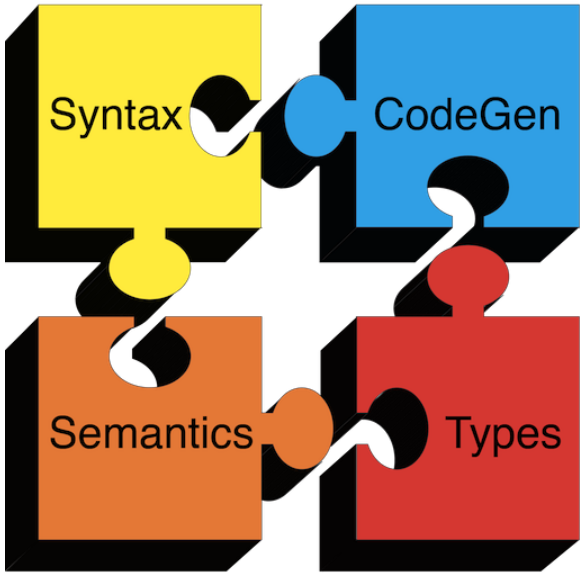


Compilers

SLR Parsing

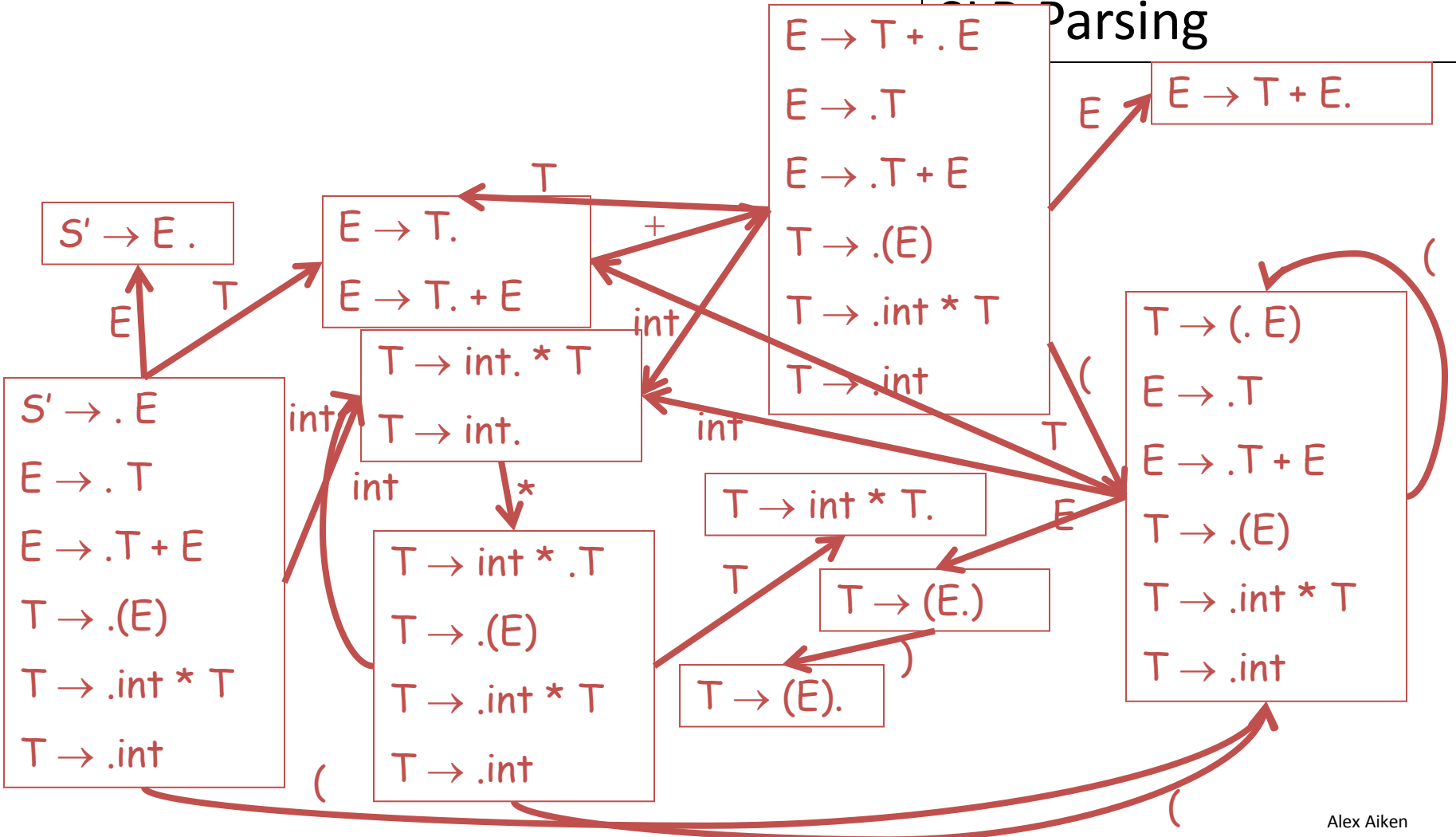
SLR Parsing

- LR(0) Parsing: Assume
  - stack contains α
  - next input is t
  - DFA on input α terminates in state s
- Reduce by $X \rightarrow \beta$ if
  - s contains item $X \rightarrow \beta.$
- Shift if
  - s contains item $X \rightarrow \beta.t\omega$
  - equivalent to saying s has a transition labeled t

SLR Parsing

- LR(0) has a reduce/reduce conflict if:
  - Any state has two reduce items:
  - $X \to \beta.$ and $Y \to \omega.$
- LR(0) has a shift/reduce conflict if:
  - Any state has a reduce item and a shift item:
  - $X \to \beta.$ and $Y \to \omega.t\delta$

Parsing


S' → E .
S' → . E
E → . T
E → .T + E
T → .(E)
T → .int * T
T → .int
E → T.
E → T. + E
T → int. * T
T → int.
T → int * .T
T → .(E)
T → .int * T
T → .int
E → T + . E
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E → T + E.
T → int * T.
T → (E.)
T → (E).
T → (. E)
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E
T
+
int
*
(
)

SLR Parsing

- SLR = “Simple LR”
- SLR improves on LR(0) shift/reduce heuristics
  - Fewer states have conflicts

SLR Parsing

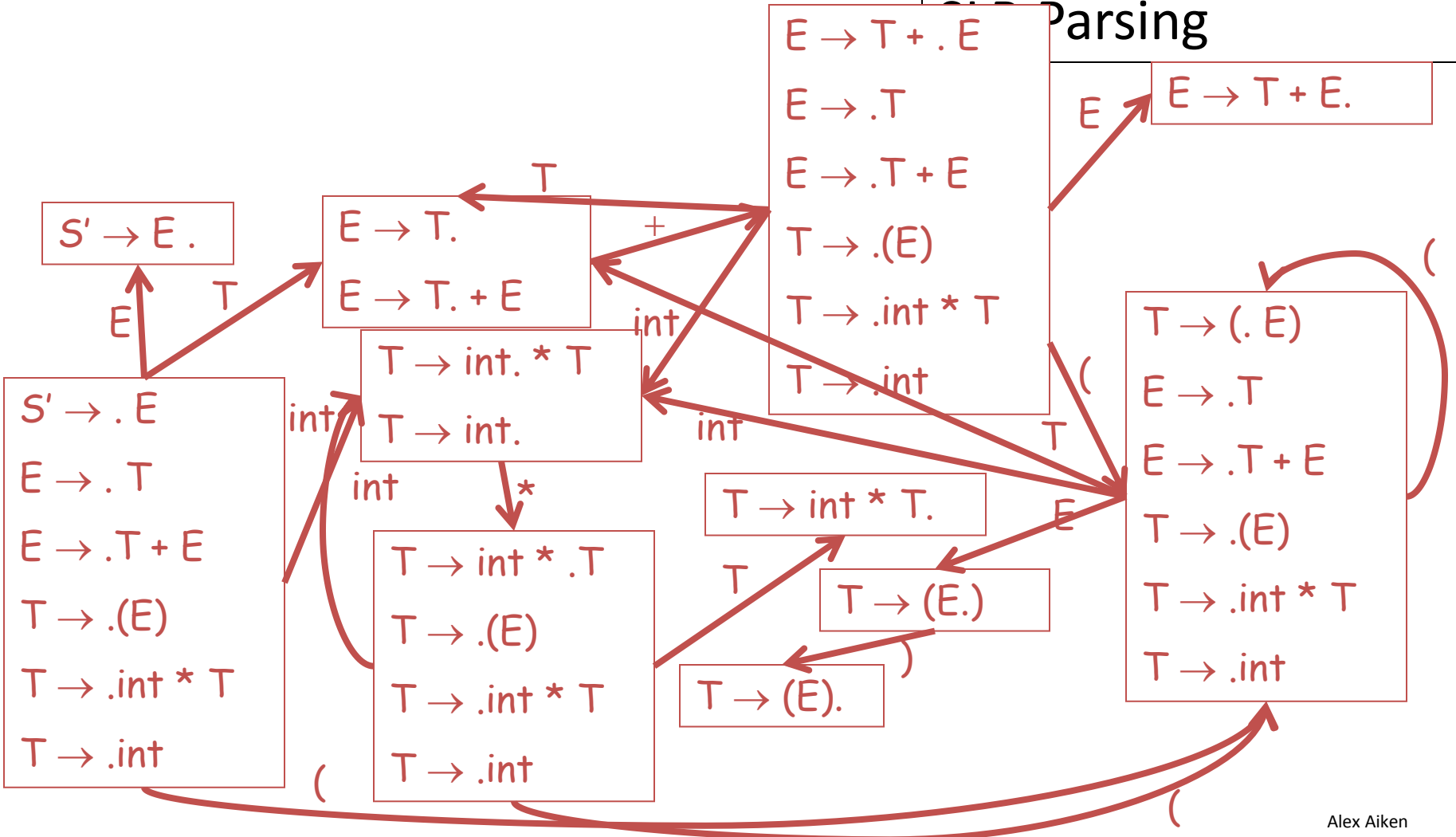
- Idea: Assume
  - stack contains α
  - next input is t
  - DFA on input α terminates in state s
- Reduce by $X \rightarrow \beta$ if
  - s contains item $X \rightarrow \beta.$
  - $t \in Follow(X)$
- Shift if
  - s contains item $X \rightarrow \beta.t\omega$

SLR Parsing

- If there are conflicts under these rules, the grammar is not SLR

- The rules amount to a heuristic for detecting handles
  - The SLR grammars are those where the heuristics detect exactly the handles

Parsing


S' → E .
S' → . E
E → . T
E → .T + E
T → .(E)
T → .int * T
T → .int
E → T.
E → T. + E
T → int. * T
T → int.
T → int * .T
T → .(E)
T → .int * T
T → .int
E → T + . E
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E → T + E.
T → int * T.
T → (E.)
T → (E).
T → (. E)
E → .T
E → .T + E
T → .(E)
T → .int * T
T → .int
E
T
+
int
*
(
)

SLR Parsing

- Lots of grammars aren't SLR
  - including all ambiguous grammars
- We can parse more grammars by using precedence declarations
  - Instructions for resolving conflicts

SLR Parsing

- Consider our favorite ambiguous grammar:
  - $E \rightarrow E + E \mid E * E \mid (E) \mid int$
- The DFA for this grammar contains a state with the following items:
  - $E \rightarrow E * E .$ $E \rightarrow E . + E$
  - shift/reduce conflict!
- Declaring "* has higher precedence than +" resolves this conflict in favor of reducing

SLR Parsing

- The term “precedence declaration” is misleading

- These declarations do not define precedence; they define conflict resolutions
  - Not quite the same thing!

SLR Parsing

1. Let M be DFA for viable prefixes of G
2. Let $|x_1 \ldots x_n\$$ be initial configuration
3. Repeat until configuration is $S|\$$
   - Let $\alpha|\omega$ be current configuration
   - Run M on current stack α
   - If M rejects α, report parsing error
     - Stack α is not a viable prefix
   - If M accepts α with items I, let a be next input
     - Shift if $X \rightarrow \beta.\ a\ \gamma \in I$
     - Reduce if $X \rightarrow \beta. \in I$ and $a \in \text{Follow}(X)$
     - Report parsing error if neither applies

SLR Parsing

- If there is a conflict in the last step, grammar is not SLR(k)

- k is the amount of lookahead
  - In practice k = 1